

Counting in Practical Anonymous Dynamic Networks is Polynomial

Maitri Chakraborty¹, Alessia Milani², and Miguel A. Mosteiro¹

¹ Kean University, Union, NJ, USA,
`{chakraborty,mmosteiro}@kean.edu`

² LABRI, University of Bordeaux, INP, Talence, France,
`milani@labri.fr`

Abstract. Anonymous Dynamic Networks is a harsh computational environment due to changing topology and lack of identifiers. Computing the size of the network, a problem known as Counting, is particularly challenging because messages received cannot be tagged to a specific sender. Previous works on Counting in Anonymous Dynamic Networks do not provide enough guarantees to be used in practice. Indeed, they either compute only an upper bound on the network size that may be as bad as exponential, or guarantee only double-exponential running time, or do not terminate, or guarantee only eventual termination without running-time guarantees. Faster experimental protocols do not guarantee the correct count.

Recently, we presented the first Counting protocol that computes the exact count with exponential running-time guarantees. The protocol requires the presence of one leader node and knowledge of any upper bound Δ on the maximum number of neighbors that any node will ever have. In the present work, we complement the latter theoretical study evaluating the performance of such protocol in practice. We tested a variety of network topologies that may appear in practice, including extremal cases such as trees, paths, and continuously changing topologies. We also tested networks that temporarily are not connected. Our simulations showed that the protocol is polynomial for all the inputs tested, paving the way to use it in practical applications where topology changes are predictable. The simulations also provided insight on the impact of topology changes on information dissemination.

To the best of our knowledge, this is the first experimental study that shows the possibility of computing the exact count in polynomial time in a variety of Anonymous Dynamic Networks that are worse than expected in practice.

Keywords: anonymous dynamic networks, counting, time-varying graphs.

We thank David Joiner for assisting us in using the Kean Terascale Cluster (KTC) for our simulations.

1 Introduction

Recently, a restrictive Anonymous Dynamic Network model where node identifiers are not available and topology changes frequently has attracted a lot of attention. With respect to topology changes, the Anonymous Dynamic Network model is well motivated by mobility and unreliable communication environments. With respect to node identifiers, although they are usually available in present networks (or labels³ are defined at startup), in future massive networks it may be necessary or at least convenient to avoid them to facilitate mass production.

In particular, the seemingly simple problem of *Counting* the number of nodes is challenging in Anonymous Dynamic Networks. Indeed, when two nodes communicate, it is not known whether they have communicated previously or not, which makes difficult to count. However, Counting is a fundamental problem in distributed computing because the network size is used to decide termination of protocols.

The literature on Counting in Anonymous Dynamic Networks (cf. [5–7, 12, 13]) focuses on distributed protocols for broadcast networks in slotted-time scenarios, assuming adversarially that topology may change completely all the time. Fruitful results obtained so far showed that Counting is feasible in Anonymous Dynamic Networks, paving the way to understand the cost of anonymity in Dynamic Networks, but it is still not known whether those protocols are practical. Indeed, the protocol in [12] computes only an upper bound on the network size that may be as bad as exponential. The protocols in [5] compute the exact count, but one guarantees only double-exponential running time and the other, called *unconscious*, does not terminate. Another protocol [6] is shown to have eventual termination, but without running-time guarantees.

On the practical side, in [7] the *unconscious* protocol [5] is augmented with a termination heuristic that allows the leader to decide when to stop. They show experimentally that the algorithm converges to a decision in a number of rounds that is linear in the size of the system. Unfortunately, it ensures a correct count only on dense graphs. In the worst case this approach could yield an arbitrarily wrong estimate, yielding a wrong overall count (“... *however, the error rate becomes high when we consider sparse and extremely disconnected graph instances or regular topologies.* ” [7]).

Recently, we presented the first Counting protocol that computes the exact count with exponential running-time guarantees in [13]. The protocol, called INCREMENTAL COUNTING, achieves a speedup over its predecessors by trying candidate sizes incrementally. The analysis of INCREMENTAL COUNTING in [13] also exposed the bottleneck for further speedup. Indeed, INCREMENTAL COUNTING and previous protocols include a *collection* phase where nodes disseminate some value in a gossip-based fashion. Under adversarial topology changes, the best theoretical upper bound known for such collection is exponential, whereas the only lower bound known is the trivial lower bound for dissemination, i.e. the dynamic diameter. Moreover, even restricting to gossip-based protocols, it

³ Throughout, we use “identifiers” or “labels” indistinctively.

is not known if there exist adversarial topologies such that collection requires exponential time. Thus, whether the protocol performs better than exponential in practice is an important question.

In the present work, we complement the theoretical study in [13] evaluating thoroughly the performance of INCREMENTAL COUNTING in practice. We tested a variety of network topologies that may appear in practice, including random, best case, and worst case. In particular, we tested (1) tree topologies carefully drawn uniformly at random from the equivalence classes defined by isomorphisms, (2) star topologies, and (3) path topologies. The simulations parameters include the size of the network n , an upper bound on the number of neighboring nodes Δ , and the period of time without topology changes T , including the extremal cases when topology changes continuously and a static topology. We also tested (4) dynamic networks with random graph topologies that may be disconnected, which are relevant given that previous works [7] do not guarantee the correct answer under disconnection.

It should be noticed that, for the purpose of dissemination in this model, tree inputs are intuitively worse than graphs, the worst case being a path. In that sense, our experimental evaluation is more challenging for the algorithm than previous works [7] where the inputs considered were various versions of random graphs that are unlikely to be trees.

For all the topologies and parameter combinations evaluated, INCREMENTAL COUNTING has proven to be polynomial in our simulations. These results motivate the application of INCREMENTAL COUNTING to practical Anonymous Dynamic Networks. Indeed, our simulations show that, for the inputs tested, Δn^4 is a loose upper bound on the running time of INCREMENTAL COUNTING. Hence, for applications where the input behavior is similar (and again we emphasize that we have used inputs that are bad for INCREMENTAL COUNTING), the collection phase of INCREMENTAL COUNTING may be stopped after Δk^4 iterations, where k is the size estimate, to obtain a protocol that can be used in practice (refer to [13] for details).

Our simulations also provide insight on the impact of network dynamics in the dissemination of information by gossip-based protocols. Indeed, our results showed that, on average, network changes speed up the computation, as long as those changes are uniform throughout the network. That is, highly dynamic topologies help rather than being a challenge as in worst-case theoretical analyses.

The rest of the paper is organized as follows. After formally defining the model and the problem in Section 2, we detail the changes introduced into INCREMENTAL COUNTING to run our simulations in Section 3. The input topologies tested and the simulation platform used are presented in Sections 4 and 5. Finally, we discuss the results obtained and our conclusions in Section 6. Beyond the overview of previous work on Counting included above and the references therein, other related work may be found in a survey on Dynamic Networks and Time-varying Graphs by Casteigts et al. [3].

2 The Anonymous Dynamic Network Model and the Counting Problem

The Anonymous Dynamic Network model is defined as follows. We consider a network composed by a set V of n nodes. For reference, we define node labels $\{1, 2, \dots, n\}$. However, nodes do not have identifiers or labels that may be used in the computation. That is, nodes cannot be distinguished, except for one node ℓ that is called the *leader*. As shown in [12], Counting is not solvable in Anonymous Networks without the presence of a leader, even if the topology does not change. If a given pair of nodes $i, j \in V$ is able to communicate directly, we say that there is a *link* among them, and we say that i and j are *neighbors*.

The communication proceeds in synchronous *rounds* through broadcast in symmetric links. That is, at each round, a node i broadcasts a message to its neighbors and simultaneously receives the messages broadcast in the same round by all its neighbors. Then, each node makes some local computation (if any).

As customary in the Anonymous Dynamic Networks literature, we assume that the time taken by computations is negligible with respect to communication. Thus, to evaluate performance, we count the number of communication rounds to complete the computation.

The set of links among nodes at round r is denoted as $E(r)$. For any given node i , we denote the set of neighbors of i at round r as $N(i, E(r))$. If the particular round is clear from context we will refer to the set of neighbors simply as $N(i, E)$ or $N(i)$ indistinctively. We assume that there is an upper bound on the size of the neighborhood of any node that is known to all nodes. That is, there is a value $\Delta \leq n - 1$ that may be used by the protocol such that, for any round r and any node $i \in V$, it is $N(i, E(r)) \leq \Delta$. Indeed, Δ is used in INCREMENTAL COUNTING as described in Section 3. Other than the upper bound Δ , nodes do not have any other information of the network topology and/or dynamics. Thus, INCREMENTAL COUNTING does not use any other information.

Previous work [5, 13] also assume knowledge of this upper bound on the neighborhood. This is because it was conjectured in [12] that any non-trivial computation is impossible without knowledge of some network characteristics. This conjecture has been recently disproved [4]. On the other hand, the algorithm used to prove the result has exponential time and space complexity.

The set of links is dynamic. That is, they may change from one round to another. In [5, 12, 13], Counting protocols have been analyzed assuming that in each round a new set of links may be chosen adversarially, as long as the network is connected. This model was presented in [10] as 1-interval connectivity model. Our simulations showed that if a new set of links is chosen uniformly at random for each round, the dissemination of information towards the leader is indeed faster than if changes are less frequent. Hence, for our simulations we generalize the connectivity model assumed in [5, 12, 13] as follows. We say that the network is *T-stable* if, after a topology change, the set of links does not change for at least T rounds. More formally, for any pair of rounds r_i, r_j such that $0 < r_i < r_j$, $E(r_i) \neq E(r_j)$, and for any $r_i \leq r_k < r_j$ it is $E(r_i) = E(r_k)$, then $r_j - r_i \geq T$. In

contrast, in T -interval connected networks it is assumed that for *any* sequence of T rounds, there is a stable set of links spanning all nodes.

Notice that for connected networks both models are the same for $T = 1$, but for $T > 1$ on tree topologies (most of our inputs), T -stable networks restrict less the adversary than T -interval connectivity networks. Indeed, if the topology is always a tree, which is a worst case scenario for dissemination, T -interval connectivity enforces a static network, whereas T -stability allows to change the tree every T rounds. In this work, we study T -stable networks and we evaluate a range of values for T , from $T = 1$ up to a static network.

We complete the section with the definition of the problem in [13]: “An algorithm is said to solve the *Counting* problem if whenever it is executed in a Dynamic Network comprising n nodes, all nodes eventually terminate and output n .”

3 Incremental Counting Protocol Simulator

INCREMENTAL COUNTING [13] is a distributed protocol that relies on the presence of a leader node. Thus, the protocol includes algorithms for the leader and non-leader nodes. Both algorithms are composed by a sequence of synchronous iterations. In each iteration the candidate size is incremented and checked to decide whether it is correct or not.

Each of the iterations in INCREMENTAL COUNTING is divided in three phases: collection, verification, and notification. In the collection phase, nodes are initially assigned a unit value, called energy. Then, iteratively nodes disseminate a fraction of their energy towards the leader. The collection phase terminates when the leader has collected enough energy to know that if the current guess of the system size is correct, then there is no node in the system with residual energy greater than a given threshold. In the verification phase, nodes inform the leader whether some node has energy above the threshold, which would mean that the candidate size is wrong. Should the candidate size be correct, in the last phase all nodes are notified that the computation is complete. Each of these phases is composed by a fixed number of communication rounds so that the synchronization of the distributed computation is given. The number of rounds of each phase is a function of the candidate size.

INCREMENTAL COUNTING runs for a fixed number of rounds for each phase. Given that the upper bound on the number of rounds needed for each phase proved in [13] is exponential, a simulation of INCREMENTAL COUNTING as in [13] would yield exponential time. The purpose of our simulations in the present work is to evaluate whether such upper bound is loose in practice. So, rather than running each phase for a fixed number of communication rounds, we do it until a condition suited for each phase is violated (cf. Algorithm 1), and we count the number of communication rounds to complete the computation. Consequently, our simulator is necessarily centralized to check such condition, but again, these changes are introduced to obtain experimentally a tighter upper bound on practical inputs, rather than to provide a practical implementation of INCREMENTAL

COUNTING. As explained before, a practical distributed INCREMENTAL COUNTING protocol must be implemented as in [13], that is, executing each phase for a fixed number of rounds, but our simulations provide a polynomial bound on that number.

In the following paragraphs, we provide further details on the changes applied to each phase of INCREMENTAL COUNTING, and how each phase is implemented in our simulator.

During the collection phase of INCREMENTAL COUNTING non-leader nodes are initially assigned a unit of energy, which is later disseminated towards the leader using a gossip-based approach [1, 8, 9]. That is, each non-leader node repeatedly shares a fraction of its energy with each neighbor. Given that the leader keeps all the energy received, it eventually collects most of the energy in the system. In the original INCREMENTAL COUNTING protocol, for each candidate size k , the number of rounds for sharing energy is fixed to a function $\tau(k)$ that has not been proven to be sub-exponential in the worst case. Thus, to evaluate whether in practice a polynomial number of rounds is enough, in our simulations we iterate the energy transfer until the conditions needed for the verification phase are met. (Refer to Lines 4 to 9 in Algorithm 1.) That is, until the leader has collected an amount of energy such that, if its guess is correct, non-leader nodes have transferred almost all their energy, i.e. all non-leader nodes have residual energy smaller than or equal to $1/k^{1.01}$.

In the Anonymous Dynamic Network model communication is carried out in rounds. In each round, every node receives from each neighboring node simultaneously. To simulate this exchange in the collection phase we follow the approach used to analyze gossip-based protocols [1, 8, 9]. That is, the energy sharing process is simulated by a multiplication of the vector of energies by a matrix of fractions shared. More precisely, we maintain a vector $\mathbf{e}_r = (e_{r_1}, e_{r_2}, \dots, e_{r_n})$ of energies, where e_{r_i} is the energy of node i at the beginning of round r . Additionally, for each input graph with set of links $E(r)$, we define a matrix $\mathbf{F}(E(r)) = (f_{ij})$, where f_{ij} is the fraction of energy of node j that node i receives. We will refer to this matrix as $\mathbf{F}(E)$ when the round number is clear from context. For example, for a complete graph with set of links E where node 1 is the leader it is

$$\mathbf{F}(E) = \begin{pmatrix} 1 & \frac{1}{2\Delta} & \frac{1}{2\Delta} & \frac{1}{2\Delta} & \dots \\ 0 & 1 - \frac{n-1}{2\Delta} & \frac{1}{2\Delta} & \frac{1}{2\Delta} & \dots \\ 0 & \frac{1}{2\Delta} & 1 - \frac{n-1}{2\Delta} & \frac{1}{2\Delta} & \dots \\ 0 & \frac{1}{2\Delta} & \frac{1}{2\Delta} & 1 - \frac{n-1}{2\Delta} & \dots \\ \vdots & \vdots & \vdots & \vdots & \ddots \end{pmatrix} \quad (1)$$

Then, the energy of nodes is iteratively computed as $\mathbf{e}_{r+1} = \mathbf{F} \cdot \mathbf{e}_r^T$. Notice that the matrix \mathbf{F} has to be changed each time the input graph changes.

During the verification phase of INCREMENTAL COUNTING non-leader nodes disseminate towards the leader the value of the maximum energy held by any non-leader node. If the residual energy of some node is greater than the above threshold, the current candidate size is deemed incorrect by the leader. To guarantee that the leader receives from all nodes, all non-leader nodes iteratively

broadcast and update the maximum energy heard, starting from their own. This phase does not tolerate disconnection of the network, since then some nodes might not be heard by the end of the loop. To evaluate disconnected topologies, in our simulations we continue the iteration until the leader has received from all nodes. (Refer to Lines 10 to 24 in Algorithm 1.)

If the candidate size was found to be correct in the verification phase, a halting message is broadcast throughout the network in the notification phase. To synchronize the computation, the notification phase of INCREMENTAL COUNTING runs for a fixed number of rounds, independently of whether the current candidate size is correct or not. As with the verification phase, the notification phase does not tolerate disconnection of the network. To evaluate disconnected topologies, in our simulations we continue the iteration also for the notification phase, in this case until all nodes receive from the leader. (Refer to Lines 25 to 37 in Algorithm 1.)

To handle disconnection, the approach followed in the latter two phases is centralized. Nevertheless, the running times obtained are not better than in INCREMENTAL COUNTING, since the verification and notification phases run for at least the same number of rounds.

In our simulations the synchronization and bookkeeping are centralized rather than distributed, and nodes are assigned labels for reference. Nevertheless, the running times obtained may be used to fix the number of rounds needed to run the protocol distributedly in a practical Anonymous Dynamic Network, should the input behavior be known.

4 Input Topologies

To thoroughly evaluate the performance of INCREMENTAL COUNTING, we have produced different topologies that may appear in practice. The parameters of our simulations include the size of the network n , the upper bound on the number of neighbors Δ , and the period of stable topology T .

We evaluate random tree topologies rooted at the leader node, drawn uniformly as described below. As extremal cases of a tree topology, we also evaluated a star rooted at the leader node and a path where the leader is the end point. We also consider Erdos-Renyi random graphs, for which we additionally parameterize the probability p that any given pair of nodes are neighbors. Although graphs have better conductance than the trees underlying them, and consequently graphs achieve convergence faster for gossip-based protocols [15], we evaluate the latter inputs for consistency with previous works. Random graphs may be disconnected, but in our simulator INCREMENTAL COUNTING has been modified to handle disconnected components. (Refer to Section 3 for details.)

We evaluated the performance of INCREMENTAL COUNTING for all values of $n \in [3, 75]$, and $T \in \{1, 10, 20, 40, 80, 160, 320, 640, 1280, \infty\}$, where $T = \infty$ corresponds to a static network. For the star and path topologies, we permute the labels every T rounds, since no other change is possible given that the topology is fixed. On the other hand, for random trees and random graphs, we produce a

Algorithm 1: Centralized simulation of leader and non-leader nodes running INCREMENTAL COUNTING [13] on T -stable topologies. V is the set of nodes and E is the set of links. Node 1 is the leader node. The other node labels are used for reference only, but not used to make decisions. $N(i, E)$ is the set of neighbors of node i according to E . $\mathbf{F}(E)$ is the matrix of fractions shared according to E (cf. Equation 1).

```

1  $k \leftarrow 1$ ,  $IsCorrect \leftarrow \text{false}$ ,  $r \leftarrow 1$ ,  $E \leftarrow$  new set of links
2 while  $\neg IsCorrect$  do
3    $k \leftarrow k + 1$ ,  $IsCorrect \leftarrow \text{true}$ 
4   Collection Phase:
5      $(e_1, e_2, e_3, \dots, e_n) \leftarrow (0, 1, 1, \dots, 1)$  // vector of energies
6     while  $e_1 < k - 1 - 1/k^{1.01}$  do
7        $(e_1, e_2, \dots, e_n) \leftarrow \mathbf{F}(E) \cdot (e_1, e_2, \dots, e_n)^T$  // broadcast simulation
8       if  $r \equiv 0 \pmod T$  then  $E \leftarrow$  new set of links
9        $r \leftarrow r + 1$ 
10  Verification Phase:
11    if  $e_1 > k - 1$  then  $IsCorrect \leftarrow \text{false}$ 
12     $(e'_1, e'_2, e'_3, \dots, e'_n) \leftarrow (0, e_2, e_3, \dots, e_n)$  // vector of max energy heard
13    for  $1 + \lceil k/(1 - 1/k^{1.01}) \rceil$  iterations do
14      for each  $i$  do  $e''_i \leftarrow \max_{j \in N(i, E) \cup \{i\}} e'_j$  // broadcast simulation
15       $(e'_1, e'_2, \dots, e'_n) \leftarrow (e''_1, e''_2, \dots, e''_n)$ 
16      if  $r \equiv 0 \pmod T$  then  $E \leftarrow$  new set of links
17       $r \leftarrow r + 1$ 
18    // for  $G_{n,p}$  input only:
19    while not all nodes “heard” by leader do
20      for each  $i$  do  $e''_i \leftarrow \max_{j \in N(i, E) \cup \{i\}} e'_j$  // broadcast simulation
21       $(e'_1, e'_2, \dots, e'_n) \leftarrow (e''_1, e''_2, \dots, e''_n)$ 
22      if  $r \equiv 0 \pmod T$  then  $E \leftarrow$  new set of links
23       $r \leftarrow r + 1$ 
24    if  $e'_1 > 1/k^{1.01}$  then  $IsCorrect \leftarrow \text{false}$ 
25  Notification Phase:
26     $(h_1, h_2, h_3, \dots, h_n) \leftarrow (IsCorrect, \text{false}, \text{false}, \dots, \text{false})$  // vector of
    halt flags
27    for  $k$  iterations do
28      for each  $i$  do  $h'_i \leftarrow \bigvee_{j \in N(i, E)} h_j$  // broadcast simulation
29       $(h_1, h_2, \dots, h_n) \leftarrow (h'_1, h'_2, \dots, h'_n)$ 
30      if  $r \equiv 0 \pmod T$  then  $E \leftarrow$  new set of links
31       $r \leftarrow r + 1$ 
32    // for  $G_{n,p}$  input only:
33    while  $\neg(h_1 \Rightarrow \bigwedge_{i \in V} h_i)$  do
34      for each  $i$  do  $h'_i \leftarrow \bigvee_{j \in N(i, E)} h_j$  // broadcast simulation
35       $(h_1, h_2, \dots, h_n) \leftarrow (h'_1, h'_2, \dots, h'_n)$ 
36      if  $r \equiv 0 \pmod T$  then  $E \leftarrow$  new set of links
37       $r \leftarrow r + 1$ 
38  output  $k$ 

```

whole new network every T rounds, since the topology is less restricted (trees) or it is not restricted at all (graphs). For stars and random graphs, we fixed the degree upper bound $\Delta = n - 1$. For the former because that is the maximum degree and for the latter to guarantee a uniform draw. For random trees and paths we used $\Delta = 2^i$, for all values of $i > 0$ such that $2^i \leq n - 1$. For random graphs we evaluated $p \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$. All our results were computed as the average over 100 executions of the protocol.

To produce our random rooted unlabeled trees we used the algorithm RANRUT presented in [14], which was proved to provide a uniform distribution on the equivalence classes defined by isomorphisms. The tree so obtained may have maximum degree larger than Δ . When that is the case, we prune the tree moving subtrees downwards until all nodes have at most Δ neighbors. This procedure increases (or does not change) the longest path to the leader, which increases (or does not change) the running time of INCREMENTAL COUNTING. Thus, with respect to a uniform distribution on rooted unlabeled trees of maximum degree Δ , our input distribution is biased “against” INCREMENTAL COUNTING providing stronger guarantees.

In contrast, topologies obtained connecting pairs of nodes stochastically until the graph is connected, usually have relatively low diameter, which may speed up the energy transfer. Algorithm 2 summarizes the rooted tree network generator used in our simulations.

Algorithm 2: Random tree generator algorithm. Auxiliary functions in Algorithm 3.

```

1 Function GENTREE( $n, \Delta$ )
2    $t \leftarrow \text{SIZES}(n)$  // Compute number of unlabeled rooted trees of size
    $1, 2, \dots, n$ .
3    $p \leftarrow \text{DISTRIB}(t, n)$  // Compute distributions on subtrees for each  $n$ .
4    $\text{tree} \leftarrow \text{RANRUT}(p, n)$  // Choose an unlabeled rooted tree uniformly at
   random.
5   PRUNE ( $\text{tree}, \Delta$ ) // Move subtrees downwards until max degree of tree
   is  $\Delta$ .
6   return tree

```

5 Simulation Platform

We developed our own simulator and input generator implementing Algorithms 1 and 2 in Java 8. The simulations were carried out on a cluster facility at Kean University known as Puma. Puma is a 130 node, 1040 core Dell cluster running Red Hat Enterprise Linux and Rocks+ 4.3. Each node has 2 quad core 2.6 GHz Xeon processors, 16GB RAM, and 350GB local storage, and are connected with Gigabit ethernet.

Algorithm 3: Auxiliary functions for Algorithm 2. Refer to [14] for details on SIZES, DISTRIB, and RANRUT.

```

1 Function SIZES( $n$ )
2   Let  $t[1 \dots n]$  be a new array
3    $t[1] \leftarrow 1, t[2] \leftarrow 1$ 
4   for  $i = 3$  to  $n$  do
5     //  $t_i \leftarrow \sum_{j=1}^{\infty} \sum_{d=1}^{\infty} \frac{dt_{i-jd}t_d}{i-1}$ 
6      $t[i] \leftarrow 0$ 
7     for  $j = 1$  to  $n$  do
8       for  $d = 1$  to  $n$  do
9         if  $jd < i$  then  $t[i] \leftarrow t[i] + d \cdot t[i - jd] \cdot t[d]$ 
10       $t[i] \leftarrow t[i]/(i - 1)$ 
11   return  $t$ 

12 Function DISTRIB( $t, n$ )
13   Let  $p[1 \dots n][1 \dots n][1 \dots n]$  be a new 3D array
14   for  $k = 3$  to  $n$  do
15     for  $j = 1$  to  $n$  do
16       for  $d = 1$  to  $n$  do
17         if  $jd < k$  then  $p[k][j][d] \leftarrow d \cdot t[k - jd] \cdot t[d]/((k - 1) \cdot t[k])$ 
18         else  $p[k][j][d] \leftarrow 0$ 
19   return  $p$ 

20 Function RANRUT( $p, n$ )
21   if  $n \leq 2$  then  $\text{tree} \leftarrow$  new tree of size  $n$ 
22   else
23     with probability distribution  $p[n][j][d]$  draw a pair  $(j, d)$ 
24      $\text{tree} \leftarrow$  RANRUT( $p, n - jd$ )
25     for  $j$  times do
26        $\text{tree}' \leftarrow$  RANRUT( $p, d$ )
27       attach  $\text{tree}'$  to the root of  $\text{tree}$ 
28   return  $\text{tree}$ 

29 Procedure PRUNE( $\text{tree}, \Delta$ )
30   while root of tree has more than  $\Delta$  neighbors do
31     detach the rightmost subtree  $\text{tree}'$  from the root of  $\text{tree}$ 
32     ATTACH ( $\text{tree}, \text{tree}', \Delta$ )
33   foreach subtree  $\text{tree}'$  of the root of tree do
34     PRUNE ( $\text{tree}', \Delta$ )

35 Procedure ATTACH( $\text{tree}, \text{tree}', \Delta$ )
36   if root of tree has less than  $\Delta$  neighbors then
37     attach  $\text{tree}'$  to the root of  $\text{tree}$ 
38   else
39     choose a subtree  $\text{tree}''$  of the root of  $\text{tree}$  uniformly at random
40     ATTACH ( $\text{tree}'', \text{tree}', \Delta$ )

```

Notice that our simulator is centralized. That is, rather than using the cluster to simulate the communication among nodes, we simply run various instances of the simulator in parallel to speedup our experiments. Using the cluster to simulate the communication among nodes would not have provided any additional insight, given that in the Anonymous Dynamic Network model communication is reliable and our protocol is deterministic.

6 Discussion

For all the topologies and parameter combinations evaluated, INCREMENTAL COUNTING has proved to be polynomial. We plot here only a subset of our results for succinctness. Consider for instance Figure 1, where we plot the number of rounds to complete the computation (log scale) as a function of the network size, for various values of degree upper bound Δ , interval of stability T , and probability p of being connected in the random graph. We also plot the function Δn^4 to contrast the growth of such polynomial function with the results obtained. It can be seen that all our results indicate a rate of growth asymptotically smaller than Δn^4 . (The upper bound could be tightened but we choose a loose one for clarity.) For small network sizes, random graphs introduce additional delays due to disconnection, but as the network scales the dynamic topology overcomes the effect of disconnections.

For the tree and path topologies the degree upper bound Δ was checked for various values. The performance of the protocol with respect to Δ for those inputs is illustrated in Figure 2. For paths, the running time increases with Δ . This is expected given that, although nodes change positions, the topology is always a path. Δ is an upper bound, hence it may be increased, but the only impact is a reduction on the fraction of energy shared with neighbors (see Algorithm 1). Such reduction applies to the energy transferred towards the leader as well as away from the leader. Nevertheless, the balance yields a slower dissemination towards the leader in the collection phase. The same behavior can be observed for the tree topology. This is somewhat surprising because we would expect a larger Δ to yield a shallower random tree. However, the impact of a smaller fraction of energy shared dominates. Notice also in Figure 2 that these observations apply whether the topology changes frequently or not.

Figure 3 shows the performance of the protocol for random graph inputs. As expected, when the probability p of any given pair of nodes being connected is low, the time needed to complete the computation grows drastically if the network does not change frequently, because some nodes cannot reach the leader. Indeed, even for 10-stable networks the performance when $p = 0.1$ is much worse than for higher values of p when the network is likely to be connected. But if we consider less-dynamic networks, such as 320-stable, the performance for $p = 0.1$ is much worse than others even for small networks.

Figure 4 illustrates the performance for different intervals of stability and representative cases of each type of input. It can be observed that when the network is highly dynamic (10-stable) the random tree and random graph produce

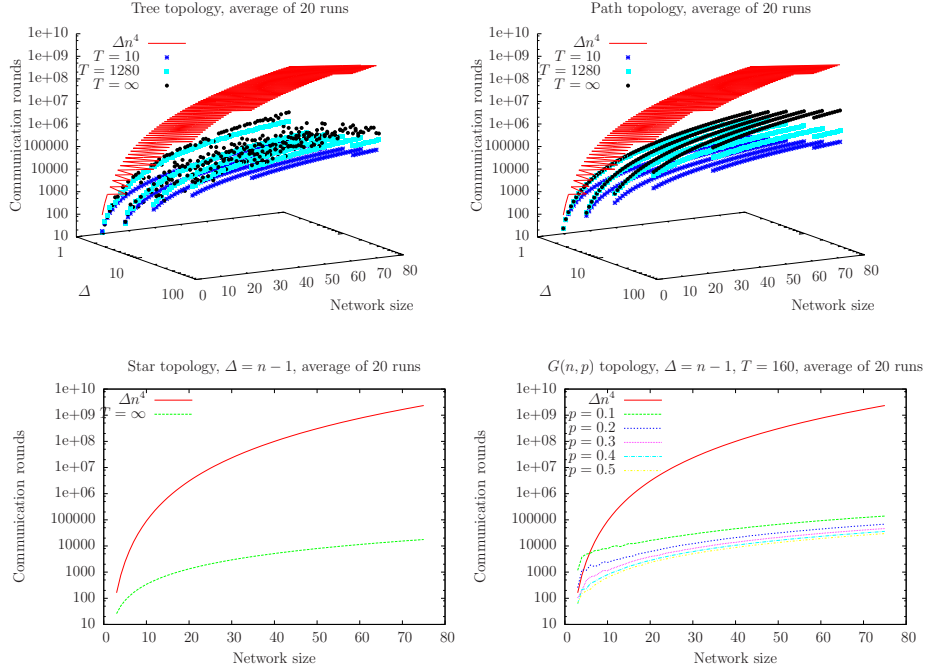


Fig. 1. INCREMENTAL COUNTING time performance compared with a polynomial function.

similar results. To understand why, consider for instance $n = 25$. Changing the topology every 10 rounds yields a network where, on average over the roughly 8000 rounds, every pair of nodes is connected. The difference between tree and graph becomes bigger as the topology changes less frequently, when the impact of the lower conductance of the tree becomes dominant.

For static random trees, we can observe also in Figure 4 that the running times varied significantly from one network size to another. This is due to the particular trees that happened to be drawn, which sometimes have better conductance than others. The random graph was not evaluated for static networks since permanent disconnection would stop the protocol from completing the computation. On the other hand, for the star input the only dynamics introduced was a permutation of labels, which are not used by INCREMENTAL COUNTING and therefore has no impact on the running time. Thus, we illustrate the results only in the static topology plots. As expected, a star yields the fastest running time since all nodes are connected to the leader.

Finally, Figure 4 shows also the performance on a path input. Given that the topology is fixed to a path, as with the star topology, the only dynamics introduced is a permutation of labels. Then, as expected, the running time increases significantly as the network changes less frequently. The reason being that in a

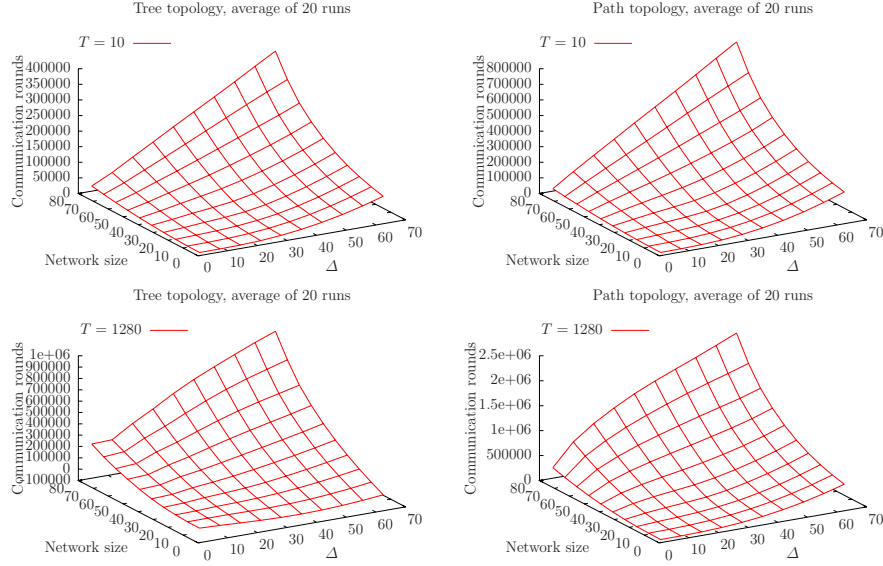


Fig. 2. INCREMENTAL COUNTING time performance vs. degree upper bound.

path where most of the nodes do not change much their distance to the leader, the fraction of energy received by the leader in each round from distant nodes is inverse exponential on that distance.

In summary, our simulations have shown that INCREMENTAL COUNTING runs in polynomial time for all the networks evaluated. The set of inputs tested comprise average topologies likely to appear in practice as well as extremal cases. Our simulations showed the static path to have the worse time performance among all inputs tested. (For illustration, see in Figure 5 the case of a path with $\Delta = 4$ for various values of T .) Intuition on why INCREMENTAL COUNTING would be polynomial in static paths can be obtained as follows. To bound the collection time, consider the inverse gossip-based process of computing the average starting with the leader having energy n and all other nodes 0. This process can be modeled as a Markov chain, whose convergence time bounds the time to compute the average (hence, the collection time). It is known [11] that the convergence time of such Markov chain can be bounded by the mixing time of a random walk on the same graph, which in turn has been proven [2] to be at most quadratic on the number of nodes for paths. However, the question of whether a static path is a worst case for Counting in some model of Anonymous Dynamic Network remains open.

As a byproduct, our simulations provided insight on the impact of network dynamics in the dissemination of information by gossip-based protocols. Indeed, our results showed that on average network changes speed-up convergence. That is, as long as the effect is uniform throughout the network, highly dynamic

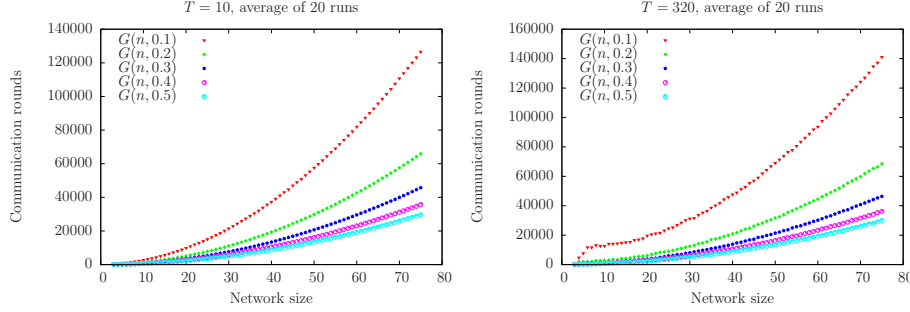


Fig. 3. INCREMENTAL COUNTING time performance on random graphs for different intervals of stability.

topologies help rather than being a challenge as in a worst-case theoretical analysis.

7 Conclusions and Open Problems

The problem of Counting in Anonymous Dynamic Networks is challenging because lack of identifiers and changing topology make difficult to decide if a new message has been received before from the same node. Counting protocols [5, 6, 13] overcome this challenge checking candidate sizes iteratively until the correct size is obtained. However, the worst-case convergence time of the gossip-based process used in those protocols has been bounded only exponentially, yielding only exponential guarantees for INCREMENTAL COUNTING [13]. The situation is even worse in other works where a huge overestimate (proved in [12]) is initially used as the candidate size, yielding a double-exponential protocol [5]. Other Counting protocols do not terminate [5], or terminate but do not provide running-time guarantees [6], or compute only an exponential upper bound on the network size [12]. Other heuristic protocols do not guarantee that the computation is correct [7].

The main goal of this work was to show that in practice INCREMENTAL COUNTING may compute the size of Anonymous Dynamic Networks in a polynomial number of rounds of communication, obtaining always the correct result. Additionally, the extensive simulations carried out provided very interesting observations about the impact of topology and dynamics in performance, as detailed in Section 6. To the best of our knowledge, this is the first experimental study of the practical time complexity of a Counting protocol that computes the exact count in Anonymous Dynamic Networks.

Open problems left for future work include further validating the conclusions in this work by simulating INCREMENTAL COUNTING using real traces, taken for instance from cellular networks, or by carrying out experiments in real networks. Also, once the network size is known, studying more complex problems

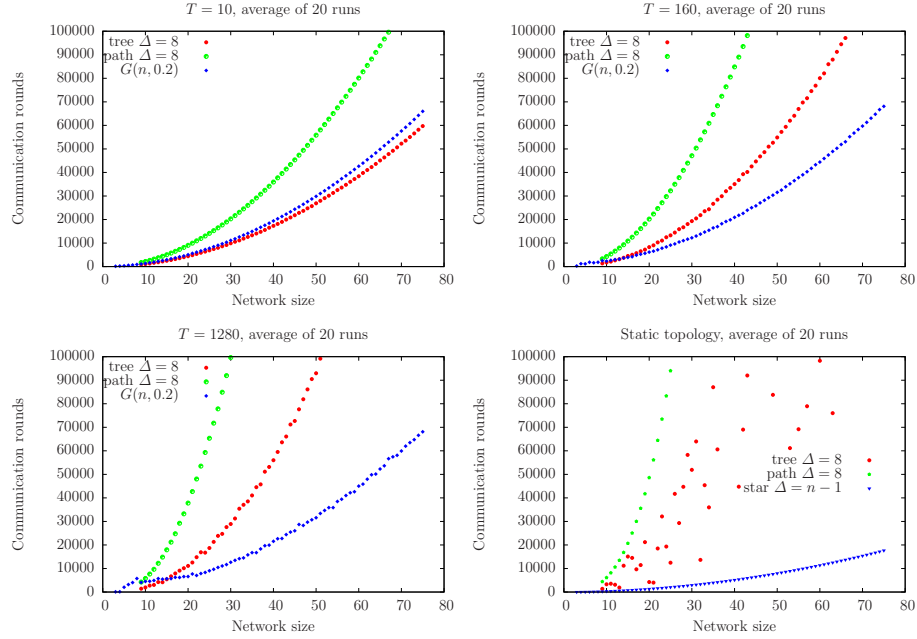


Fig. 4. INCREMENTAL COUNTING time performance for different intervals of stability.

in Anonymous Dynamic Networks is the natural next step. On the theoretical side, proving a polynomial upper bound on the time for counting remains an enticing open question.

References

1. Almeida, P.S., Baquero, C., Farach-Colton, M., Jesus, P., Mosteiro, M.A.: Fault-tolerant aggregation: Flow-updating meets mass-distribution. In: Proceedings of the 15th International Conference on Principles of Distributed Systems. Lecture Notes in Computer Science, vol. 7109, pp. 513–527. Springer (2011)
2. Beveridge, A., Wang, M.: Exact mixing times for random walks on trees. *Graphs and Combinatorics* 29(4), 757–772 (2013)
3. Casteigts, A., Flocchini, P., Quattrociocchi, W., Santoro, N.: Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems* 27(5), 387–408 (2012)
4. Di Luna, G.A., Baldoni, R.: Non trivial computations in anonymous dynamic networks. In: Proceedings of the 19th International Conference on Principles of Distributed Systems. Leibniz International Proceedings in Informatics (2015), to appear.
5. Di Luna, G.A., Baldoni, R., Bonomi, S., Chatzigiannakis, I.: Conscious and unconscious counting on anonymous dynamic networks. In: Proceedings of the 15th International Conference on Distributed Computing and Networking. Lecture Notes in Computer Science, vol. 8314, pp. 257–271. Springer Berlin Heidelberg (2014)

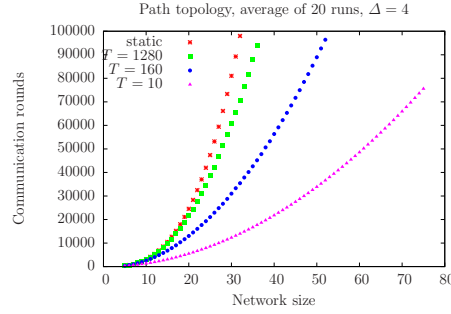


Fig. 5. INCREMENTAL COUNTING time performance for a path topology.

6. Di Luna, G.A., Baldoni, R., Bonomi, S., Chatzigiannakis, I.: Counting in anonymous dynamic networks under worst-case adversary. In: Proceedings of the 34th International Conference on Distributed Computing Systems. pp. 338–347. IEEE (2014)
7. Di Luna, G.A., Bonomi, S., Chatzigiannakis, I., Baldoni, R.: Counting in anonymous dynamic networks: An experimental perspective. In: Proceedings of the 9th International Symposium on Algorithms and Experiments for Sensor Systems, Wireless Networks and Distributed Robotics. Lecture Notes in Computer Science, vol. 8243, pp. 139–154. Springer Berlin Heidelberg (2014)
8. Fernández Anta, A., Mosteiro, M.A., Thraves, C.: An early-stopping protocol for computing aggregate functions in sensor networks. *J. Parallel Distrib. Comput.* 73(2), 111–121 (2013), <http://dx.doi.org/10.1016/j.jpdc.2012.09.013>
9. Kempe, D., Dobra, A., Gehrke, J.: Gossip-based computation of aggregate information. In: Proc. of the 44th IEEE Ann. Symp. on Foundations of Computer Science. pp. 482–491 (2003)
10. Kuhn, F., Lynch, N., Oshman, R.: Distributed computation in dynamic networks. In: Proceedings of the 42nd ACM Symposium on Theory of Computing. pp. 513–522. ACM (2010)
11. Levin, D.A., Peres, Y., Wilmer, E.L.: Markov chains and mixing times. American Mathematical Soc. (2009)
12. Michail, O., Chatzigiannakis, I., Spirakis, P.G.: Naming and counting in anonymous unknown dynamic networks. In: Stabilization, Safety, and Security of Distributed Systems, pp. 281–295. Springer (2013)
13. Milani, A., Mosteiro, M.A.: A faster counting protocol for anonymous dynamic networks. In: Proceedings of the 19th International Conference on Principles of Distributed Systems. Leibniz International Proceedings in Informatics (2015), to appear.
14. Nijenhuis, A., Wilf, H.S.: Combinatorial Algorithms for Computers and Calculators: 2d Ed. Academic Press (1978)
15. Sinclair, A., Jerrum, M.: Approximate counting, uniform generation and rapidly mixing markov chains. *Information and Computation* 82(1), 93–133 (1989)